

Class 7: Machine Learning 1

Samuel Fisher (A18131929)

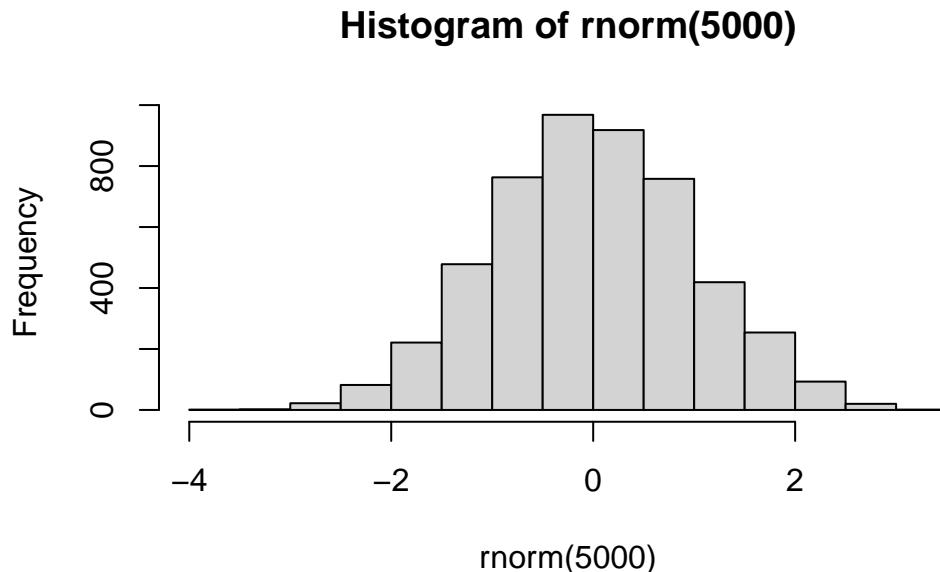
Background

Today we will begin our exploration of some important machine learning methods, namely **clustering** and **dimensionality reduction**.

Let's make up some input data where for clustering where we know what the natural "clusters" are

The function `rnorm()` can be useful here

```
hist(rnorm(5000))
```



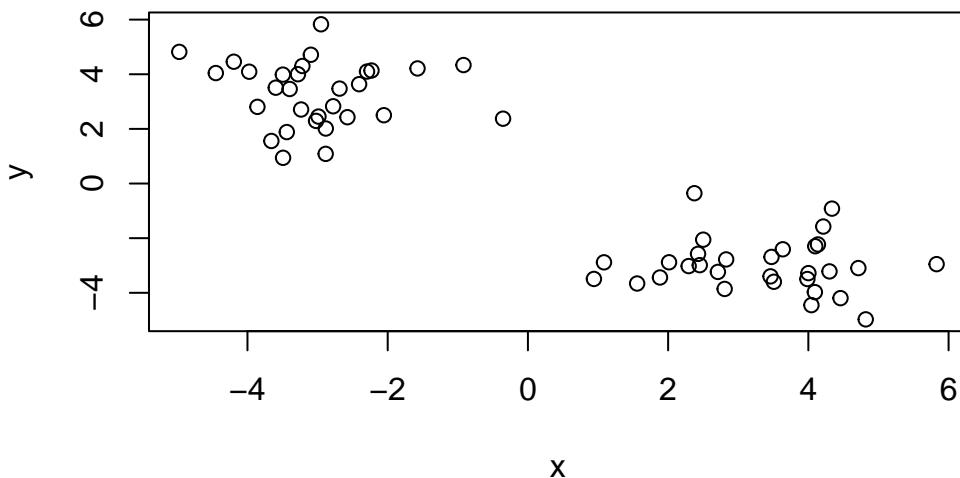
Q. Generate random numbers centered at +3

```

tmp <- c(rnorm(30, mean=3),
         rnorm(30, mean=-3))

x <- cbind(x=tmp, y=rev(tmp))
plot(x)

```



K-means clustering

The main function in “Base R” for K-means clustering is called `kmeans()`

```

km <- kmeans(x, centers = 2)
km

```

K-means clustering with 2 clusters of sizes 30, 30

Cluster means:

	x	y
1	-2.998766	3.299268
2	3.299268	-2.998766

Clustering vector:

```
Within cluster sum of squares by cluster:  
[1] 67.60563 67.60563  
  (between_SS / total_SS =  89.8 %)
```

Available components:

```
[1] "cluster"      "centers"       "totss"        "withinss"      "tot.withinss"  
[6] "betweenss"    "size"          "iter"         "ifault"
```

Q. What component of the results object details the cluster sizes?

km\$size

[1] 30 30

Q. What component of the results object details the cluster centers?

km\$center

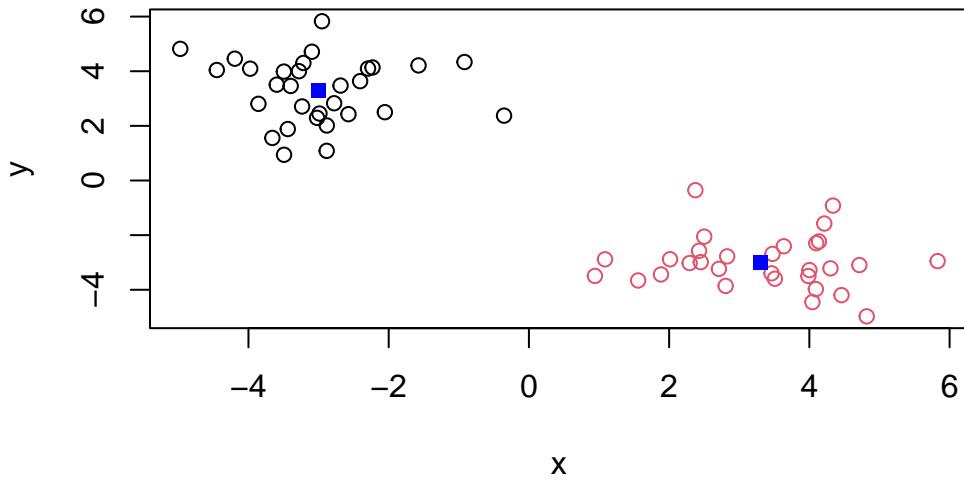
	x	y
1	-2.998766	3.299268
2	3.299268	-2.998766

Q. What component of the results object detralls the cluster membership vector (i.e. our main result of which points lie in which cluster)?

km\$cluster

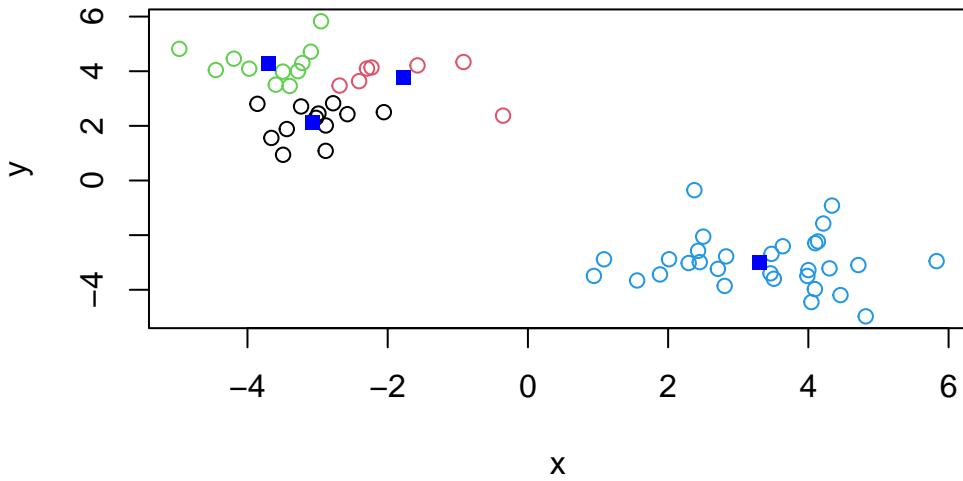
Q. Plot our clustering results with points colored by cluster and also add the cluster centers as new points colored blue

```
plot(x, col=km$cluster)  
points(km$centers, col ="blue", pch=15)
```



Q. Run `kmeans()` again and this time produce 4 clusters (and call your result object `k4`) and make a results figure like above?

```
k4 <- kmeans(x, 4)
plot(x, col=k4$cluster)
points(k4$centers, col ="blue", pch=15)
```



The metric

```
km$tot.withinss
```

```
[1] 135.2113
```

```
k4$tot.withinss
```

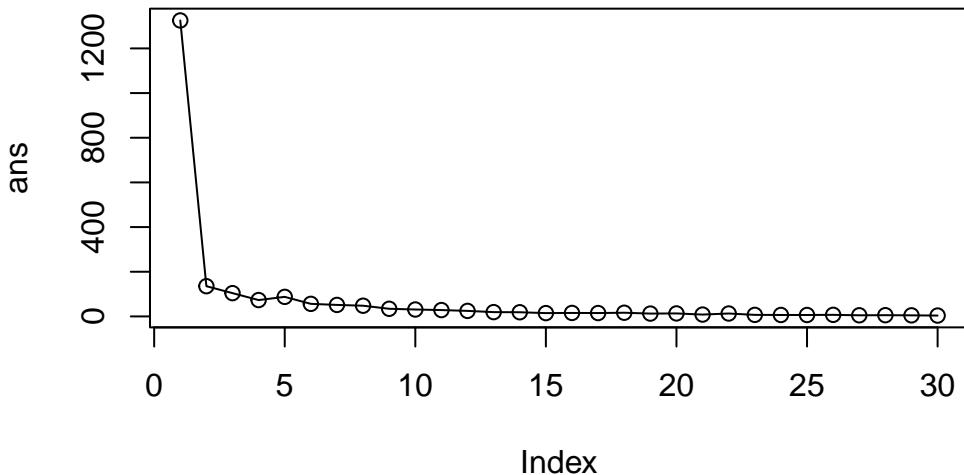
```
[1] 90.62597
```

Q. Let's try different number of K (centers) from 1 to 30 and see what the best result is

```
ans <- NULL
for(i in 1:30) {
  ans <- c( ans, kmeans(x, centers = i)$tot.withinss)
}
ans
```

```
[1] 1325.167891 135.211253 104.167860 73.124467 87.467600 56.240645
[7] 51.454001 47.871187 34.170353 30.941848 28.441936 24.660491
[13] 19.230458 18.542345 14.829878 15.553977 15.000944 16.419063
[19] 12.671023 13.290462 8.250537 12.954822 6.911918 6.417560
[25] 6.303756 7.053867 4.994505 5.458898 4.780425 3.754837
```

```
plot(ans, typ="o")
```



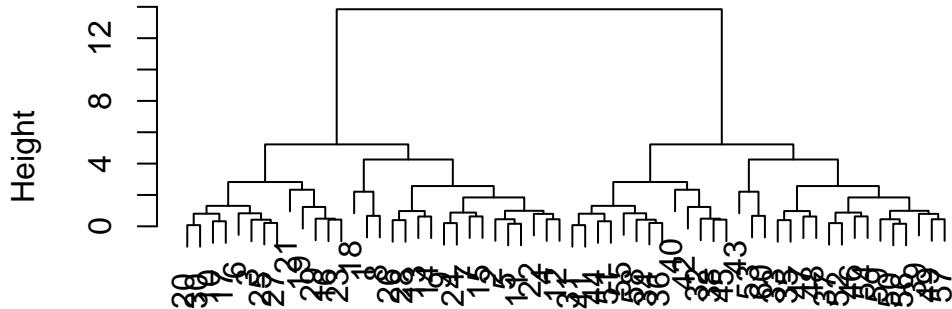
Key-point K-means will impose a clustering structure on your data even if it is not there - it will always give you the answer you asked for even if that answer is silly!

Hierarchical Clustering

The main function for Hierarchical Clustering is called `hclust()`. Unlike `kmeans()` (which does all the work for you) you can't just pass `hclust()` our raw input data. It needs a "distance matrix" like the one returned from the `dist()` function.

```
d <- dist(x)
hc <- hclust(d)
plot(hc)
```

Cluster Dendrogram

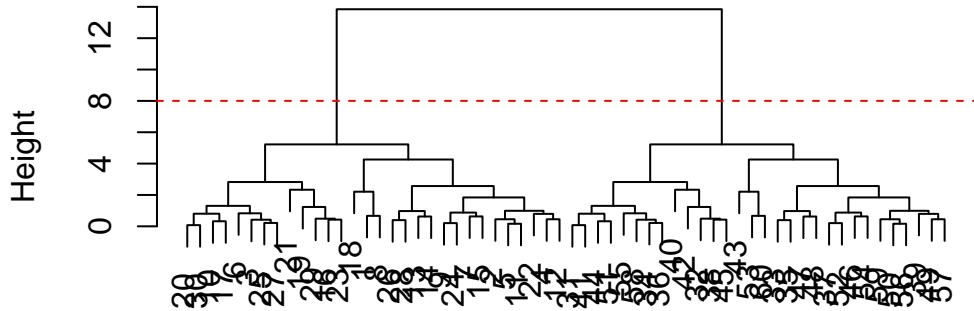


```
d  
hclust (*, "complete")
```

To extract our cluster membership vector from a `hclust()` result object we have to “cut” our tree at a given height to yield separate “groups”/“branches”.

```
plot(hc)  
abline(h=8, col="red", lty=2)
```

Cluster Dendrogram



```
d  
hclust (*, "complete")
```

To do this we use the `cutree()` function on our `hclust()` object:

```
grps <- cutree(hc, h=8)  
grps
```

```
[1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2  
[39] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

```
table(grps, km$cluster)
```

```
grps 1 2  
1 0 30  
2 30 0
```

PCA of UK food Data

Import the dataset of food consumption in the UK:

```
url <- "https://tinyurl.com/UK-foods"  
x <- read.csv(url)  
x
```

		X	England	Wales	Scotland	N.Ireland
1	Cheese		105	103	103	66
2	Carcass_meat		245	227	242	267
3	Other_meat		685	803	750	586
4	Fish		147	160	122	93
5	Fats_and_oils		193	235	184	209
6	Sugars		156	175	147	139
7	Fresh_potatoes		720	874	566	1033
8	Fresh_Veg		253	265	171	143
9	Other_Veg		488	570	418	355
10	Processed_potatoes		198	203	220	187
11	Processed_Veg		360	365	337	334
12	Fresh_fruit		1102	1137	957	674
13	Cereals		1472	1582	1462	1494
14	Beverages		57	73	53	47
15	Soft_drinks		1374	1256	1572	1506
16	Alcoholic_drinks		375	475	458	135
17	Confectionery		54	64	62	41

Q. How many rows and columns are in your new data frame named x: What R functions could you use to answer this?

```
dim(x)
```

```
[1] 17 5
```

One solution to set the row names is to do it by hand...

```
rownames(x) <- x[,1]
```

To remove the first column I can use the minus index trick

```
x<- x[,-1]
```

A better way to do thids is to set the row names to the first column with `read.csv()`

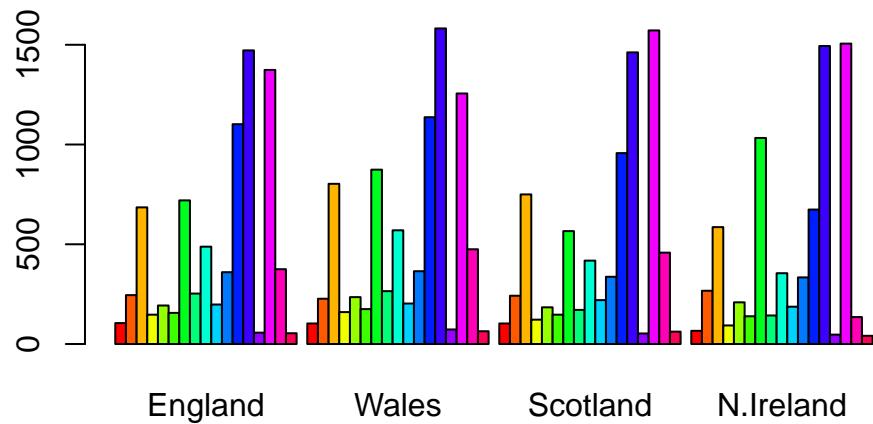
```
x <- read.csv(url, row.names = 1)
x
```

	England	Wales	Scotland	N.Ireland
Cheese	105	103	103	66
Carcass_meat	245	227	242	267
Other_meat	685	803	750	586
Fish	147	160	122	93
Fats_and_oils	193	235	184	209
Sugars	156	175	147	139
Fresh_potatoes	720	874	566	1033
Fresh_Veg	253	265	171	143
Other_Veg	488	570	418	355
Processed_potatoes	198	203	220	187
Processed_Veg	360	365	337	334
Fresh_fruit	1102	1137	957	674
Cereals	1472	1582	1462	1494
Beverages	57	73	53	47
Soft_drinks	1374	1256	1572	1506
Alcoholic_drinks	375	475	458	135
Confectionery	54	64	62	41

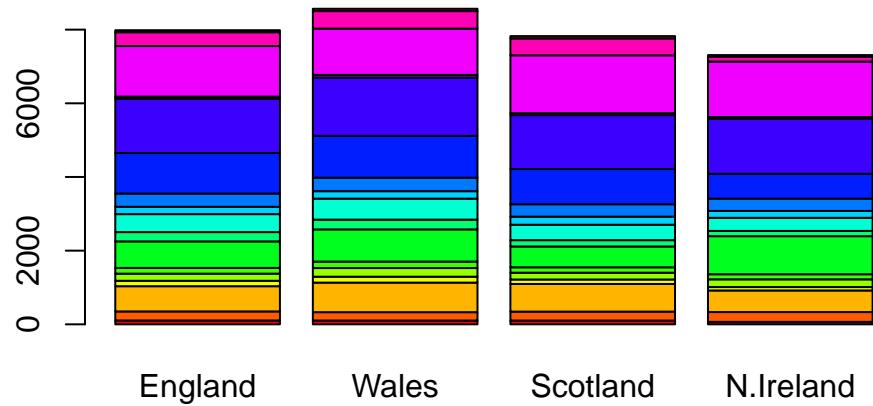
Spotting major differences and trends

Is difficult even in this wee 17D dataset...

```
barplot(as.matrix(x), beside=T, col=rainbow(nrow(x)))
```

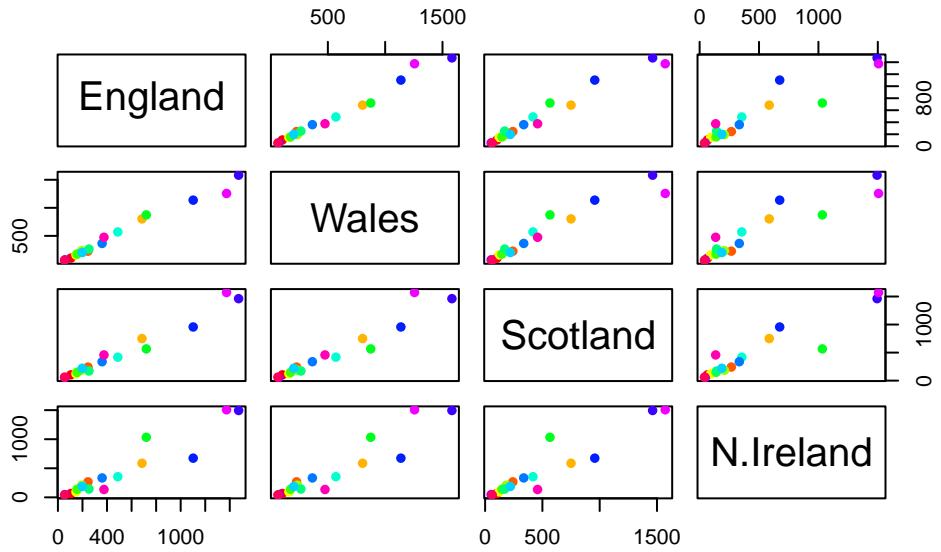


```
barplot(as.matrix(x), beside=F, col=rainbow(nrow(x)))
```

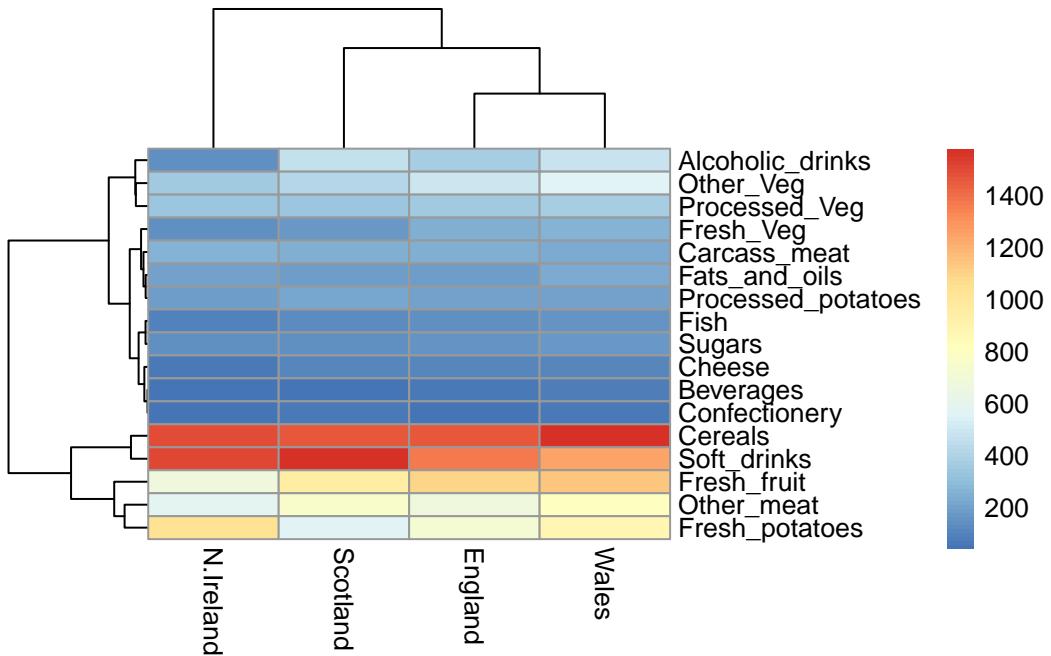


Pairs plots and heatmaps

```
pairs(x, col=rainbow(nrow(x)), pch=16)
```



```
library(pheatmap)
pheatmap(as.matrix(x))
```



PCA to the rescue

The main PCA function in “base R”

```
pca <- prcomp(t(x))
```

```
summary(pca)
```

Importance of components:

	PC1	PC2	PC3	PC4
Standard deviation	324.1502	212.7478	73.87622	2.7e-14
Proportion of Variance	0.6744	0.2905	0.03503	0.0e+00
Cumulative Proportion	0.6744	0.9650	1.00000	1.0e+00

```
attributes(pca)
```

```
$names
[1] "sdev"      "rotation"   "center"    "scale"     "x"

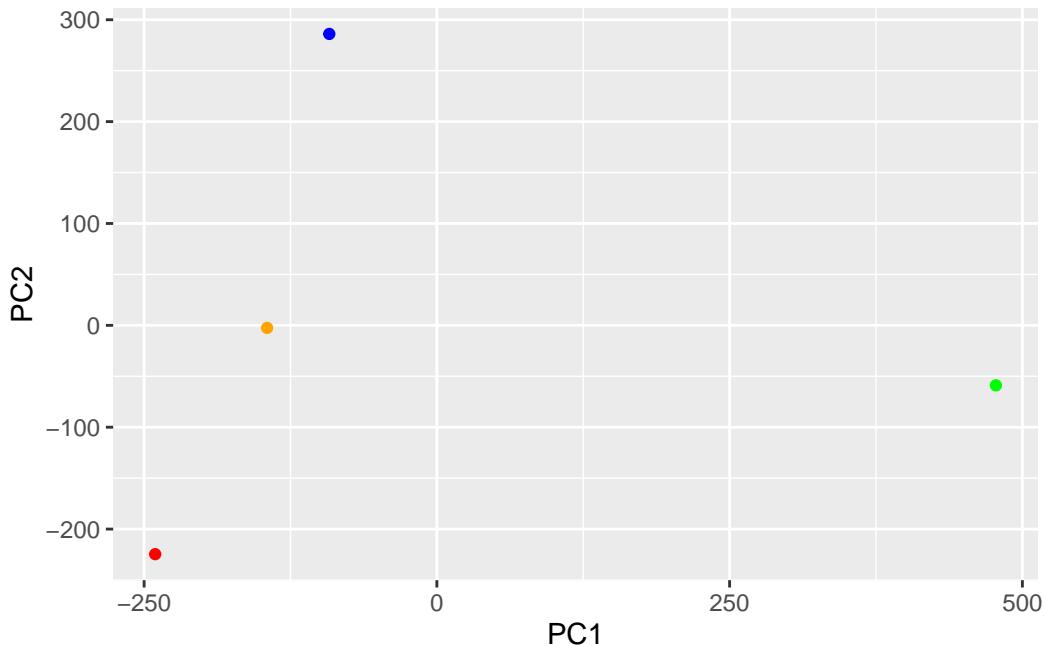
$class
[1] "prcomp"
```

To make one of main PCA result figure we turn to `pca$x` the scores along our new PCs. This is called “PC Plot” or “Score Plot” or “Ordination Plot”, etc.

```
my_cols <- c("orange", "red", "blue", "green")
```

```
library(ggplot2)

ggplot(pca$x) +
  aes(PC1, PC2) +
  geom_point(col= my_cols)
```



The second major result figure is called a “loadings plot” of “variable contributions plot” or “weight plot”

```
ggplot(pca$rotation) +
  aes(x= PC1, y= reorder(rownames(pca$rotation), PC1)) +
  geom_col(fill = "steelblue")
```

